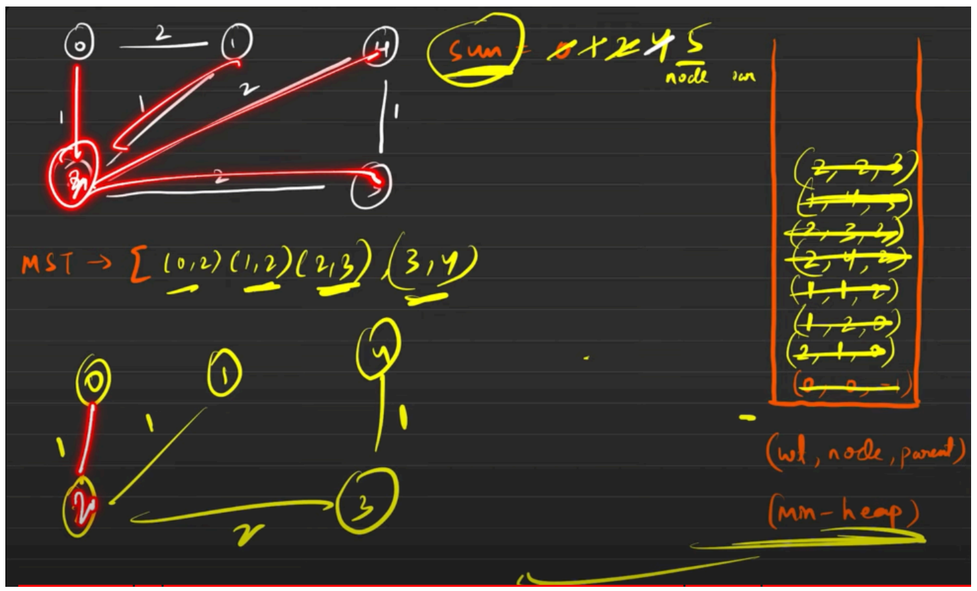
Prims is very similar to BFS/Dijkstras. However, we should not mark elements as visited as soon as we reach it, rather only when we use it in top and add it's weight.

This is also kind of a greedy algorithm

```java
class Solution {
    static int spanningTree(int V, int E, List<List<int[]>> adj) {
        // Code Here.
        boolean[] visited = new boolean[V];
        PriorityQueue<Pair> pq = new PriorityQueue<>((a, b) -> Integer.compare(a.weight, b.weight));
        pq.offer(new Pair(0, 0));

        int res = 0;
        while (!pq.isEmpty()) {
            int curNode = pq.peek().node, curWeight = pq.poll().weight;

            if (visited[curNode])
                continue;

            visited[curNode] = true;
            res += curWeight;

            for (int[] ng: adj.get(curNode)) {
                if (visited[ng[0]])
                    continue;
                pq.offer(new Pair(ng[0], ng[1]));
            }
        }

        // buildGraph(adj, V);

        return res;
    }

    private static class Pair {
        int node;
        int weight;
        // int parent;

        Pair (int node, int weight) {
            this.node = node;
            this.weight = weight;
        }

    }
}
```



Time complexity : O (E log E + E log E)
The first E log E comes from traversing the nodes in PQ using while loop and other E logE for second for loop.
Space complexity: O (E)